Robotica (1997) volume 15, pp 293–304. Printed in the United Kingdom © 1997 Cambridge University Press

On-line learning control of manipulators based on artificial neural network models*

**M. Kemal Ciliz and †Can Işik

(Received in Final Form: July 2, 1996)

SUMMARY
This paper addresses the tracking control problem of robotic manipulators with unknown and changing dynamics. In this study, nonlinear dynamics of the robotic manipulator is assumed to be unknown and a control scheme is developed to adaptively estimate the unknown manipulator dynamics utilizing generic artificial neural network models to approximate the underlying dynamics. Based on the error dynamics of the controller, a parameter update equation is derived for the adaptive ANN models and local stability properties of the controller are discussed. The proposed scheme is simulated and successfully tested for trajectory following tasks. The controller also demonstrates remarkable performance in adaptation to changes in manipulator dynamics.

KEYWORDS: Neural network; On-line learning; Tracking control; Manipulators.

1 INTRODUCTION

Trajectory following control of robotic manipulators has been an important research area in the last decade. Highly nonlinear and coupled dynamics of the manipulators hinder the efficient use of well known linear control techniques. Although reasonable trajectory following performance can be achieved using linearized models of the manipulators, nonlinear model based (computed torque and feedforward control) methods still remain as the most efficient control schemes for trajectory tracking problems.[1] However these methods assume the existence of an exact stationary parametric model of the manipulator dynamics.

In the case of changing dynamics and/or unknown dynamics, various adaptive control techniques are possible. These usually assume a reduced order simple dynamic model of the manipulator[2] or a sophisticated parametric model with an adaptation algorithm estimating the unknown model parameter.[3,4] Craig et al.[3,4] formulated a nonlinear parametric model-based adaptive controller based on Lyapunov's stability theory. Slotine and Li took a slightly different approach[5] where they derive a globally convergent parameter update law based on the passivity properties of the manipulator dynamics and the Lyapunov's stability theory. A tutorial paper by Ortega and Spong[6] gives an overall review on the recent developments in adaptive control of manipulators.

Note that all the adaptive algorithms which are cited in this section make use of a parametric model, either a simple one or a more sophisticated one. Hence the uncertainties involved in these cases are considered as structured uncertainties.

In some cases, however, the control system designer may not have a parametric dynamic model of the system and the uncertainties. In this case, a control methodology known as *learning control* has been introduced originally by Arimoto et al.[7] These schemes are commonly referred to as *trajectory learning* schemes due to the repetitive nature of these algorithms over one specific trajectory. Another important research area in trajectory following control is the *table look-up* method. These tabularization methods have been suggested originally by Albus[8] under the name Cerebellar Model Articulation Controller (CMAC) and extended in the works of Raibert[9] and Miller et al.[10]

With the resurgence of artificial neural network (ANN) research in recent years for various problems of nonlinear nature, many researchers have attempted to apply neurologically inspired algorithms for manipulator control. The main underlying assumption in these applications is the efficient capability of ANNs to approximate multivariable functions. Researchers from different disciplines have published extensively on the use neuromorphic models for the control of nonlinear dynamic systems. Here we quote some of the work,[11–16] which we believe, are most representative of the research efforts in this field.

In neuromorphic robotic control approaches, most of the discussion is on the choice of the error signal to drive the parameter update dynamics. If an ANN model is used to acquire the manipulator's forward dynamics, the choice of error signal is rather trivial (the difference between the system output and the ANN model output). When ANN models are placed in a controller architecture for approximating inverse dynamics of the manipulator, it is claimed that the teaching signals for ANNs can not be derived a priori. In this respect, Kawato et al.[13] used the scaled output position and velocity error signals for training which they tried to publicize under the name feedback error learning. Based on an error equation derived from closed loop system dynamics, Ciliz and Işik[17] also used the scaled position

* This work was in part sponsored by Westinghouse Education Foundation under contract No:3597262.
** Electrical Engineering Department, Boğaziçi University Bebek, Istanbul 80815 (Turkey).
† Electrical and Computer Engineering Department, Syracuse University, Syracuse, NY 13244-1240 (USA).

and velocity error signals as the driving term for the update dynamics.

The present paper extends our previous results[17,18] in the more general framework of closed loop error dynamics of adaptive tracking controllers. Generic multilayer ANN models are utilized as parametric models of each joint's dynamics. Each neural net structure is viewed as a nonlinear extension of a deterministic auto-regressive model which is commonly used in model matching problems for linear systems. Closed loop system's error dynamics is obtained and using sliding control concepts an error surface is defined. Then a cost function is generated as the squared error distance to this surface. Based on the error dynamics, output tracking error bounds are derived using operator algebra techniques. Adaptation properties of the controller is also discussed and simulation studies were consistent with our recent results on the stability and convergence of localized closed loop dynamics.[19]

Proposed controller differs from some of the previously cited off-line learning techniques in the sense that adaptation algorithm is computed on-line using the operational data of actual arm movements based on the given desired trajectory. Whereas in off-line techniques, ANN models are first trained for identification of system dynamics and then placed in a controller architecture.[15]

The layout of the present paper is as follows. First basic properties of multilayer ANN models are introduced. Section 3 presents the trajectory following problem and discusses the proposed learning control architecture for manipulators. In this section, the error dynamics equation is generated, and the parameter update equation is derived. Based on the closed-loop system dynamics, a local stability analysis is given which is quoted from a recent work by the authors.[19] Tracking error bounds of the controller are also computed in this section. In Section 4 we give the simulation results of the controller for trajectory following and adaptation tasks. In Section 5, results and the practical advantages of the scheme are discussed.

2 ARTIFICIAL NEURAL NETWORKS FOR SYSTEM MODELING

A multilayer artificial neural network (ANN) model is basically a nonlinear extension of a linear adaptive model (i.e. a model whose output is defined as a linear combination of its input variables). Such architectures received renewed interest in recent years with the introduction of new learning paradigms in the works of Werbos[20] and Rumelhart et al.[21] A generic multilayer ANn model is illustrated in Figure 1. In a multilayer ANN model, the input vector is processed through the intermediate (hidden) layers of adaptive weights of the network before reaching the output layer.

What makes ANN models important tools in nonlinear system analysis is their capability of approximating multivariable nonlinear functions. For a given set of operational data representing an arbitrary nonlinear function over a compact subset of the function's input space, the interconnections (weights) of the ANN model are adjusted in such a way that the input-output properties of the network approximates those of the underlying nonlinear function. The weight/parameter update or adjustment algorithm is based on the idea of minimizing a cost function and originally introduced by Werbos[20] and later widely publicized by Rumelhart et al.[21] as the *backpropagation* algorithm due to its structural properties. Derivation of the algorithm is rather straightforward and can be found in reference 21. Next we discuss the *nonlinear mapping* properties of ANN models in a mathematical framework.

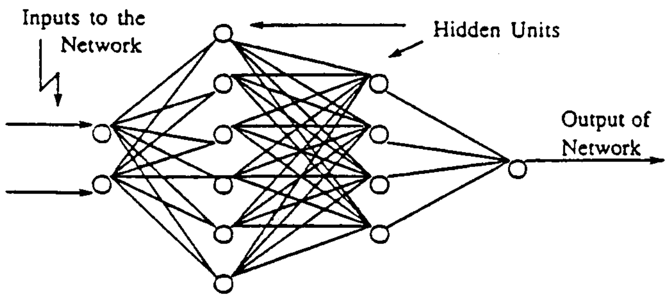

Fig. 1. Architecture of a multilayer Artificial Neural Network (ANN) Model.

2.1 Nonlinear functional approximation by ANN models

A generic neural network architecture which is illustrated in Figure 1 basically defines a mapping between two vector spaces. There has been experimental evidence that ANNs are capable of approximating arbitrary nonlinear functionals over compact subsets of their input space.[22–25] Recently a series of papers also appeared in literature stating theorems that show the validity of this claim, by Funahashi,[26] Cybenko,[27] and Hornik, Stinchcombe and White.[28] There is active research investigating the nonlinear approximation capabilities of ANN models and however there are still open questions to be answered (especially on the number of units needed to attain a given degree of accuracy in approximation). Here we quote an existence theorem by Funahashi [26] on the functional approximation property of an ANN model. Similar results are also reported in [27, 28].

Let the points of an n-dimensional normed vector space $\mathcal{R}^n$ be denoted by $X = (x_1, \ldots, x_n)$ and the L^n_∞ norm of X is defined by,

$$\|X\|_\infty = \max_i (|x_1|, \ldots, |x_n|)$$

Let K be a closed bounded subset of $\mathcal{R}^n$, and a real vector valued functions $f(\cdot)$ be defined on K, as $f: K \subset \mathcal{R}^n \to \mathcal{R}$. We would like to approximate this function by using a layered network with bounded nonlinearities as the activation functions for the hidden layers, and with linear functions for the input and output layers. Before stating the existence theorem, we write the input-output relationship of a hidden unit in a multilayer network, as $y = g(\sum_{i=1}^n w_i x_i)$ where $g(\cdot)$ is a nonconstant, bounded and monotone increasing function (e.g., a sigmoid), and w, x, and y represent the weights, inputs to the unit and output of the unit, respectively. n denotes the number of the inputs to that unit.

Theorem 2.1 *Let $g(\cdot)$ be a nonconstant, bounded and monotone increasing continuous function. Let K be a compact subset of $\mathscr{R}^n$ and $f(x_1, \ldots, x_n)$ be a real valued continuous function defined on K. Then for an arbitrary small positive $\varepsilon > 0$, there exists an integer "m" and real constants w_j, $(j = 1, \ldots, m)$, and v_{ij} $(i = 1, \ldots, n;$ $j = 1, \ldots, m)$ such that,*

$$\hat{f}(x_1, \ldots, x_n) = \sum_{j=1}^{m} w_j g\left(\sum_{i=1}^{n} v_{ii} x_i\right)$$

satisfies $\max_{X \in K} |f(x_1, \ldots, x_n) - \hat{f}(x_1, \ldots, x_n)| \leq \varepsilon$. *That is* $\|f(X) - \hat{f}(X)\|_\infty \leq \varepsilon$.

This theorem shows that for some arbitrary $\varepsilon > 0$, there exists a three layer network whose output functions for the hidden layer are $g(\cdot)$ (e.g. sigmoid), whose output functions for input and output layers are linear and which has an input output function $\hat{f}(x_1, \ldots, x_n)$ such that the approximation,

$$\max_{X \in K} |f(X) - \hat{f}(X)| = \|f(X) - \hat{f}(X)\|_\infty \leq \varepsilon \tag{1}$$

holds. The proof of the theorem is given in reference [26]. A similar result is also given by Hornik and White (Theorem 2.4 and Corollary 2.6 in reference [28]) which proves that a single hidden layer neural network can approximate any multivariable continuous function uniformly on any compact set.

The above result is quite powerful in presenting the existence of nonlinear mapping capabilities of neural networks. Simple nonlinear bounded monotone increasing continuous functions are assumed as activation functions of the nodes, hence this justifies the use of *sigmoidal nonlinearities* (i.e. $g(x) = 1/1 + e^{-x}$) in our application. The above theorem gives strong mathematical justification to utilize neural networks as nonlinear functional approximators for adaptive signal processing and control systems applications. In the rest of this paper, the mathematical analysis of the controller architecture is based on the above theorem.

Before finishing this section, we would like to emphasize that mathematically proving functional approximation properties of ANNs is still an active and growing interdisciplinary research area. However, our intent here is just to use neural networks as a tool in controller algorithms, therefore we are content with the above theoretical result.

3 THE PROPOSED CONTROLLER ARCHITECTURE

In this section we briefly discuss the trajectory following control problem of rigid mechanical manipulators, then propose an ANN based controller architecture. Consider the vector representation of an n link rigid manipulator dynamics, given as

$$\tau = M(\mathbf{q})\dot{\mathbf{q}} + \mathbf{v}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \tag{2}$$

where τ is the $n \times 1$ vector of joint torques, and $\mathbf{q}$ is the $n \times 1$ vector of joint positions. The matrix $M(\mathbf{q})$ is the $n \times n$ positive definite "inertia matrix". $\mathbf{v}(\mathbf{q}, \dot{\mathbf{q}})$ is $n \times 1$ vector function representing centrifugal and Coriolis effects, and finally $n \times 1$ vector function $\mathbf{g}(\mathbf{q})$ represents torques due to gravity. The derivation of (2) can be found in common reference texts.[1] Equation (2) can be put in a more compact form as,

$$\tau = M(\mathbf{q})\dot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \tag{3}$$

Given a bounded *desired trajectory* in joint variables $(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)$, the control designer's task is to devise a controller to track this desired trajectory as closely as possible. If exact manipulator dynamics is available, then the *control*

$$\tau = M(\mathbf{q}(K_v\dot{\mathbf{e}} + K_p\mathbf{e} + \ddot{\mathbf{q}}_d) + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \tag{4}$$

will result in error equation of the form,

$$\ddot{\mathbf{e}} + K_v\dot{\mathbf{e}} + K_p\mathbf{e} = 0 \tag{5}$$

due to the cancellation of nonlinear terms, where $\mathbf{e} = \mathbf{q}_d - \mathbf{q}$ and $\dot{\mathbf{e}} = \dot{\mathbf{q}}_d - \dot{\mathbf{q}}$. K_v and K_p are the diagonal matrices of *velocity* and position feedback gains, respectively. Note that the above error equation is a decoupled one due to the diagonal nature of the constant matrices K_p and K_v. Therefore adjusting these gain matrices properly, tracking errors can be effectively forced to zero.

The equation given in (3) represents the *inverse dynamics* of a robotic arm. That is, given a set of joint variables, $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, (3) gives the corresponding torque to drive the actuators. Based on this assertion, the manipulator's direct dynamics can be readily obtained as follows,

$$\ddot{\mathbf{q}} = M^{-1}(\mathbf{q})\tau - M^{-1}(\mathbf{q})\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \tag{6}$$

$$\dot{\mathbf{q}} = R(\mathbf{q}, \dot{\mathbf{q}}, \tau) \tag{7}$$

(Note that positive definiteness of the inertia matrix $M(\mathbf{q})$ guarantees the existence of its inverse.) The direct dynamics represented in short by $R(\mathbf{q}, \dot{\mathbf{q}}, \tau)$ actually refers to a **nonlinear transformation** (mapping) from manipulator's input (joint torques τ) to the manipulator's output (joint motion). Based on this argument and equation (7), the manipulator's inverse dynamics can then be written as,

$$\tau = R^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{8}$$

where $R^{-1}(\cdot)$ is used to denote the inverse transformation obtained by inverting the robot direct dynamics $R(\cdot)$. These nonlinear transformations are time dependent, however in the rest of the text, the arguments of R, R^{-1} and their variants will sometimes be dropped for the brevity of the analysis.

If an accurate parametric dynamic model is available, such as the one given by (3), then a computed torque or a feedforward control scheme can be efficiently utilized. However if such a model is not available, the system dynamics have to be adaptively identified in order to achieve a feedforward compensation. In such cases, learning control methods can be used to generate the necessary feedforward torques.

3.1 ANN-model based adaptive control

Here we propose the use of a generic ANN architecture to model the inverse dynamic structure of each joint.[17,18] Reexamining equation (8), this nonlinear transformation can be efficiently modeled by ANNs as discussed in Section 2 and justified mathematically by Theorem 2.1.

The manipulator's inverse dynamics, defined by the nonlinear transformation R^{-1}, can be decomposed into n transformations, namely

$$\tau = R^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \begin{bmatrix} r_1^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \\ \vdots \\ r_n^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \end{bmatrix} \tag{9}$$

where each $r_i^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, $i = 1, \ldots, n$ defines the inverse dynamics of the corresponding joint, that is, $r_i^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\colon \mathcal{R}^{3n} \to \mathcal{R}$, with $i = 1, \ldots, n$. Based on Theorem 2.1, each entry $r_i^{-1}(\cdot)$ of the vector function $R^{-1}(\cdot)$ can be modeled by an ANN such that the overall system's inverse dynamics model is represented by,

$$\tau = \hat{R}^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \begin{bmatrix} \hat{r}_1^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \\ \vdots \\ \hat{r}_n^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \end{bmatrix} = \begin{bmatrix} N_1(\mathbf{q}, \dot{\mathbf{q}}_1, \ddot{\mathbf{q}}, \mathbf{p}_1) \\ \vdots \\ N_n(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{p}_n) \end{bmatrix} \tag{10}$$

where $(\hat{\cdot})$ denotes the *estimated* models, and $N_i(\cdot)$, $i = 1, \ldots, n$ represents the output of each ANN model that is used to realize the time dependent nonlinear mapping $r_i^{-1}(t)$. $\mathbf{p}_i$ can be considered as the vector of all adjustable weights of the corresponding ANN model and will be defined explicitly in the sequel.

Here we define an augmented state vector of the robot dynamics as, $\mathbf{z}(t) = \{\mathbf{q}^T(t),\ \dot{\mathbf{q}}^T(t),\ \ddot{\mathbf{q}}^T(t)\}^T \in \mathcal{R}^k$ with $k = 3n$, which denotes a time dependent input vector of the inverse dynamics, $R^{-1}(\mathbf{z})$ **[18]**. Assuming that a three-layer ANN with n inputs, m hidden layer neurons and one output neuron is used to model each individual joint's inverse dynamics, we can explicitly write this model as,

$$\hat{r}_i^{-1}(\mathbf{z}(t)) = N_i(\mathbf{z}(t), \mathbf{w}_i(t), H_i(t)) = \mathbf{w}_i^T(t)\Upsilon(H_i(t)\mathbf{z}(t)) \tag{11}$$

where $\mathbf{w}_i(t) \in \mathcal{R}^m$ is the output layer weight (parameter) vector, $H_i(t) \in \mathcal{R}^{m \times k}$ is the hidden layer weight matrix of the "i"th ANN model. $\mathbf{z}(t) \in \mathcal{R}^k$ with $k = 3n$ is the input vector as defined before. In the rest of the text, time argument of these vectors will sometimes be dropped for the brevity of the analysis. The hidden layer vector function $\Upsilon(\cdot) \in \mathcal{R}^m$ is defined as

$$\Upsilon(\cdot) = \begin{bmatrix} g_1(\cdot) \\ \vdots \\ g_m(\cdot) \end{bmatrix} \tag{12}$$

where $g_i(\cdot) \in \mathcal{R}$ is by definition a bounded monotone increasing function which is taken to be a sigmoid function in this case, based on the justification given by the Theorem 2.1 and Theorem 2.4 of reference [28]. Hence $g_i(x) = 1/(1 + e^{-x})$ and it is bounded as $0 \le g_i(x) \le 1$. To put the hidden layer weight matrix of the ANN model in vectoral form, we define $\mathbf{v}_i = vec(H_i) \in \mathcal{R}^{mk}$ where $vec(\cdot)$ operator gives a vector which is obtained by stacking the columns of its matrix argument. Then the argument $H_i\mathbf{z}$ of vector function $\Upsilon(\cdot)$ can be written as,

$$H_i\mathbf{z} = \Phi\mathbf{v}_i \tag{13}$$

where $\mathbf{v}_i = \{H_{11}, H_{21}, \ldots, H_{m1}, \ldots, H_{1k}, \ldots, H_{mk}\}^T$ and $\Phi \in \mathcal{R}^{m \times nk}$ is a matrix which can be considered as the modified input of the ANN model and is defined as,

$$\Phi = \begin{pmatrix} z_1 & 0 & \cdots & 0 & z_2 & 0 & \cdots & 0 & \cdots & z_k & 0 & \cdots & 0 \\ 0 & z_1 & \cdots & 0 & 0 & z_2 & \cdots & 0 & \cdots & 0 & z_k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z_1 & 0 & 0 & \cdots & z_2 & \cdots & 0 & 0 & \cdots & z_k \end{pmatrix} \tag{14}$$

where $z_i \in \mathcal{R}$ are the elements of the input vector $\mathbf{z} \in \mathcal{R}^k$. Hence (11) can now be written as,

$$\hat{r}_i^{-1}(\mathbf{z}, \mathbf{w}_i, \mathbf{v}_i) = N(\mathbf{z}, \mathbf{w}_i, \mathbf{v}_i) = \mathbf{w}_o^T\Upsilon(\Phi\mathbf{v}_i) \tag{15}$$

A similar representation can be obtained for networks with more than one hidden layer. With this implicit parametric manipulator model, we can investigate the controller structure that would be suited for our application. However note that, since an ANN model basically realizes a *direct implicit transformation* from the input vector $\mathbf{z}$ to joint torques τ, this model does not convey any explicit information on manipulator's estimated dynamic components such as in the inertia matrix. Hence a direct adaptive control architecture which would be based on a computed torque-like model is **not possible**. With the assumption that a manipulator model does not exist, generic ANN models can be effectively used to approximate the manipulator dynamics. Then the feedforward torques generated by the ANN models can be combined with a feedback servo signal to obtain the torques that will finally drive the actuators. Hence, the control law can be written as,

$$\tau = \hat{R}^{-1}(\mathbf{z}) + K_v\dot{\mathbf{e}} + K_p\mathbf{e} = \mathbf{N}(\mathbf{z}) + K_v\dot{\mathbf{e}} + K_p\mathbf{e} \tag{16}$$

where $K_v \in \mathcal{R}^{n \times n}$ and $K_p \in \mathcal{R}^{n \times n}$ are the diagonal gain matrices with entries k_v and k_p, respectively, $\hat{R}^{-1}(\mathbf{z}) = \mathbf{N}(\mathbf{z}) = \{N_1, \ldots, N_n\}^T \in \mathcal{R}^n$ is the dynamic model estimate which consists of "n" ANN models which repesent the actuators' inverse dynamics. A block diagram of the proposed controller is shown in Figure 2. Note that the computation of the transformation $\hat{R}^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ in (16) requires the information on $\ddot{\mathbf{q}}$ in addition to the manipulator's state vector $\{\mathbf{q}, \dot{\mathbf{q}}\}$. The acceleration vector $\ddot{\mathbf{q}}$ can be computed by differentiating the velocity vector $\dot{\mathbf{q}}$ using a first order filter. Although this is not a desirable process due to possible side-effects such as

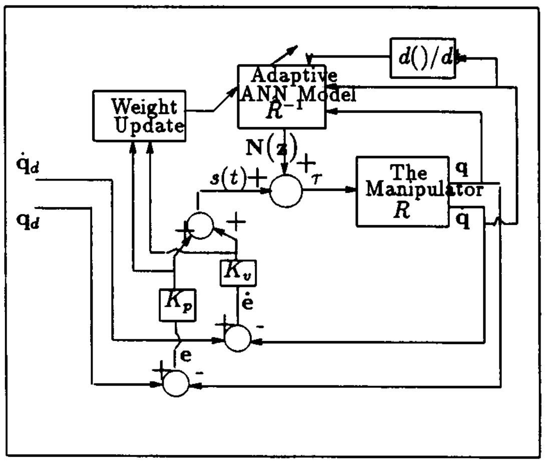

Fig. 2. Block Diagram of the Controller Architecture.

increasing susceptibility to noise, such measurements are successfully used in various adaptive control algorithms by the authors in real-time applications[29,30]

3.2 Error dynamics and derivation of parameter update equation

With the control vector given in (16), the system's error dynamics can be written by substituting (16) in (3),

$$\hat{R}^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) + K_v\dot{\mathbf{e}} + K_p\mathbf{e} = M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = R^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{17}$$

$$K_v\dot{\mathbf{e}} + K_p\mathbf{e} = R^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) - \hat{R}^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{18}$$

$$K_v\dot{\mathbf{e}} + K_p\mathbf{e} = \tilde{R}^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \tilde{R}^{-1}(\mathbf{z}) \tag{19}$$

where $\tilde{R}^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \tilde{R}^{-1}(\mathbf{z}) \in \mathcal{R}^n$ denotes the error between the actual inverse dynamics R^{-1} and the estimated model $\hat{R}^{-1}$, and can be explicitly written as,

$$\tilde{R}^{-1}(\mathbf{z}) = \begin{bmatrix} r_1^{-1}(\mathbf{z}) - \hat{r}_1^{-1}(\mathbf{z}) \\ \vdots \\ r_n^{-1}(\mathbf{z}) - \hat{r}_n^{-1}(\mathbf{z}) \end{bmatrix} = \begin{bmatrix} r_1^{-1}(\mathbf{z}) - N_1(\mathbf{z}, \mathbf{p}_1) \\ \vdots \\ r_n^{-1}(\mathbf{z}) - N_n(\mathbf{z}, \mathbf{p}_n) \end{bmatrix} \tag{20}$$

$$\tilde{R}^{-1}(\mathbf{z}) = \begin{bmatrix} \tilde{r}_1^{-1}(\mathbf{z}, \mathbf{p}_1) \\ \vdots \\ \tilde{r}_n^{-1}(\mathbf{z}, \mathbf{p}_n) \end{bmatrix} \tag{21}$$

where $\tilde{r}_i^{-1}(\mathbf{z}, \mathbf{p}_i)$ with $i = 1, \ldots, n$ denotes the error in inverse dynamic modeling for each joint, and $\mathbf{p}_i = \{\mathbf{w}_i^T, \mathbf{v}_i^T\}^T \in \mathcal{R}^{m+mk}$ is the adaptive weight vector of the corresponding ("i"th) ANN model.

Using (19) and (21), the error dynamics (with diagonal K_p and K_v matrices) for each joint can be written as follows,

$$k_p(q_{id} - q_i) + k_v(\dot{q}_{id} - \dot{q}_i) = \tilde{r}_i^{-1}(\mathbf{z}, \mathbf{p}_i), \text{ for } i = 1, \ldots, n. \tag{22}$$

$$\underbrace{k_pe_i + k_v\dot{e}_i}_{s(t)} = \tilde{r}_i^{-1}(\mathbf{z}, \mathbf{p}_i) \tag{23}$$

where e_i and $\dot{e}_i$ denote the position and velocity errors at joint i, respectively, k_p and k_v are the individual servo gains, respectively, and $\tilde{r}_i^{-1}(\mathbf{z}, \mathbf{p}_i)$ denotes the error in inverse dynamics modeling for joint "i". Here we use *sliding control concepts* and define a time varying surface $S(t)$ in the e_i and $\dot{e}_i$ space, as

$$S(t)\colon\ s(e_i, \dot{e}_i, t) = 0$$

with $s(t) = s(e_i, \dot{e}_i) = k_pe_i(t) + k_v\dot{e}_i(t)$. Hence the scalar signal $s(t)$ can be considered as the *distance* to the surface $S(t)$. The problem of tracking is then equivalent to that of minimizing the distance to the surface defined by $s(t) = 0$. If this condition is satisfied, this leads to a homogeneous differential equation $k_v\dot{e}_i + k_pe_i = 0$ whose unique solution is $e_i = 0$ and $\dot{e}_i = 0$. Examining (23), minimizing the distance to the surface $s(t) = 0$ is equivalent to minimizing the residual nonlinear error dynamics $\tilde{r}^{-1}(\cdot)$. Hence we define an instantaneous cost function $\mathcal{J}_i(t)$ for a specific sampling instant for each joint of the manipulator as,

$$\mathcal{J}_i(t) = \tfrac{1}{2}s^2(t) \tag{24}$$

This cost function gives the squared distance to the surface $s(t) = 0$. Minimizing this cost function over the weight space of the corresponding ANN model forms the basis of the **weight update algorithm.** Several minimization techniques can be employed on (24). Here we utilize a simple gradient update algorithm which can be simply written as,

$$\dot{\mathbf{p}}_i = -\alpha\nabla\mathcal{J}_i(t) \tag{25}$$

α is the adjustment gain constant (learning rate in neural-network terminology). Computing the gradient $\nabla\mathcal{J}_i(t)$ with respect to the weight vector $\mathbf{p}_i$, we get,

$$\dot{\mathbf{p}}_i = -\alpha s(t)\frac{\partial s(t)}{\partial \mathbf{p}_i} = -\alpha s(t)\frac{\partial \tilde{r}_i^{-1}(\mathbf{z}, \mathbf{p}_i)}{\partial \mathbf{p}_i} \tag{26}$$

$$\dot{\mathbf{p}}_i = -\alpha s(t)\frac{\partial(r_i^{-1}(\mathbf{z}) - N_i(\mathbf{z}, \mathbf{p}_i))}{\partial \mathbf{p}_i} \tag{27}$$

$$\dot{\mathbf{p}}_i = \alpha s(t)\frac{\partial N_i(\mathbf{z}, \mathbf{p}_i)}{\partial \mathbf{p}_i} \tag{28}$$

where $s(t)$ is as defined in (23). Computation mechanism of the gradient term in (28) is the so called backpropagation algorithm. Dropping the subscript "i" for the brevity of the analysis, we evaluate (28) explicitly for adjustable weight vectors $\mathbf{w}$ and $\mathbf{v}$, using (15) as,

$$\dot{\mathbf{w}} = \alpha s(t)\frac{\partial N}{\partial \mathbf{w}} = \alpha s(t)\frac{\partial(\mathbf{w}^T\Upsilon(\Phi\mathbf{v}))}{\partial \mathbf{w}} = \alpha s(t)\Upsilon(\Phi\mathbf{v}) \tag{29}$$

and

$$\dot{\mathbf{v}} = \alpha s(t)\frac{\partial N}{\partial \mathbf{v}} = \alpha s(t)\frac{\partial(\mathbf{w}^T\Upsilon(\Phi\mathbf{v}))}{\partial \mathbf{v}} = \alpha s(t)\Phi^TJ\mathbf{w} \tag{30}$$

where $\Phi \in \mathcal{R}^{mk\times m}$ is as defined in (14), and $J \in \mathcal{R}^{m\times m}$ is a diagonal Jacobian matrix, whose diagonal entries are given as

$$J_{jj} = \left.\frac{\partial g_j(x)}{\partial x}\right|_{x=\{\Phi \mathbf{v}\}_j}$$

Writing the above update equations in vectoral form, we get

$$\dot{\mathbf{p}} = \begin{bmatrix} \dot{\mathbf{w}} \\ \dot{\mathbf{v}} \end{bmatrix} = \alpha s(t) \underbrace{\begin{bmatrix} \Upsilon(\Phi \mathbf{v}) \\ \Phi^T J \mathbf{w} \end{bmatrix}}_{\Psi} = \alpha s(e_i, \dot{e}_i)\Psi(\mathbf{p}) \qquad (31)$$

where Ψ can be considered as a nonlinear regressor vector. The *unusual* nonlinear parametric dependence of the regressor vector is due to the nonlinearities used in the hidden layer. In this case the regressor system can not be represented linearly in terms of the adaptive weights (parameters). Therefore, the well known methods of converge analysis for the parametric-linear error dynamics can not be employed here.

Update equation (31) is in fact the backpropagation algorithm for a three layer neural network [21]. For networks with more than one hidden layer, derivation of the update equations is similar and can be realized by backpropagation. Employing the above update equation for the weight adaptation minimizes $s(t)$ and in effect the residual approximation error $\tilde{r}_i^{-1}$. Hence it effectively forces the error dynamics towards the surface, $s(e_i, \dot{e}_i) = 0$.

Note that the above update equations define a system of coupled nonlinear differential equations and this hampers a global stability analysis of the closed loop error dynamics defined by (23) and (31). However local stability properties of the closed loop system can be investigated. Since the neural network model can be not explicitly written in terms of its weights, an error equation which is linear in unknown weights can not be written. Authors recently proposed a local stability analysis of the closed loop system dynamics utilizing linearization techniques. Here we quote some of the results of our work. A detailed exposition of this analysis is given in reference 19.

3.3 *Local stability and convergence analysis*

Local stability properties of the closed loop dynamic system defined by (23) and (31) can be studied using linearization techniques. Linearization dictates that subject to smoothness of the nonlinear operators in (23) and (31), one can constitute a linearized system whose stability properties are identical to the local stability properties of the original system.

First the closed loop dynamics given by (23) and (31) is put into a state-space form as follows (subscript indexes are dropped),

$$\dot{\mathbf{x}} = A\mathbf{x} + B\tilde{r}^{-1}(\mathbf{z}, \mathbf{p}) \quad \text{and} \quad y = \mathbf{c}\mathbf{x} \qquad (32)$$

$$\dot{\mathbf{p}} = \alpha s(\mathbf{x})\Psi(\mathbf{p}) \qquad (33)$$

where $\mathbf{x}$ represents the state of equation (23), A, B and $\mathbf{c}$ are appropriate state matrices. Then $[\mathbf{x}^T, \mathbf{p}^T]^T$ forms the state vector of the closed loop system. Based on the assumption that the system is operating near a nominal state such that the ANN weights are close to their desired values and error state $\mathbf{x}$ is close to zero, the state equations given in (32) and (33) can be linearized around this operating point. The linearized system can be written as follows:[19]

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\tilde{\mathbf{p}}} \end{bmatrix} = \begin{bmatrix} A & B\Psi_* \\ -\Gamma\Psi_*^T\mathbf{c} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \tilde{\boldsymbol{p}} \end{bmatrix} \qquad (34)$$

where $\tilde{\mathbf{p}} = \mathbf{p}_* - \mathbf{p}$ is the parameter error vector with $\mathbf{p}_* = [\mathbf{w}_*, \mathbf{v}_*]$ denoting the desired weights, $\mathbf{c}$ is the output vector of the state equation (32) and Ψ_* can be considered as the *linearized regressor vector* given as,

$$\Psi_* = \begin{bmatrix} \Upsilon(\Phi_* \mathbf{v}_*) \\ \Phi_*^T J_* \mathbf{w}_* \end{bmatrix} \qquad (35)$$

where the entries of Ψ_* are evaluated at their nominal values (denoted by * signs). If the linear part of system given in (23) is strictly positive real (SPR), then based on Kalman-Yakubovich lemma, it can be shown that the tracking error vector $\mathbf{x}$ of linearized system converges to zero.[19] This basically means that, when the control system is operating in the neighbourhood of a nominal state, then the perturbations in the closed loop system dynamics can be adaptively compensated and the tracking error is forced to zero. A theorem showing this convergence property is given in reference 19.

The analysis given above can be used to investigate the adaptation properties of controller to changes in the manipulator dynamics. Simulation results of the adaptation tests are given in Section 4. Note that the above results are valid only locally. However they can be used to make qualitative statements about the closed loop system operating around a nominal state.

Even though a direct stability and convergence analysis is not possible for the nonlinear closed loop system dynamics, based on the assumption that a close approximation of inverse dynamics is achieved by ANN models, tracking error bounds can be analyzed using operator algebra techniques.

3.4 *Tracking errors bounds*

In this section, we investigate the bounds on the tracking errors due to residual nonlinearities generated by the ANN approximation of the inverse dynamics. In the following analysis bounded input conditions are assumed and operator algebra and L_∞ norms are used.

The error equation given in (19) is a linear decoupled vector differential equation with a nonlinear forcing term. We first define an operator $\mathscr{H}: \tilde{R}^{-1}(\mathbf{z}) \to \mathbf{e}$ which is infact a mapping between two n-dimensional spaces. The L_∞ gain of $\mathscr{H}$ can be compute directly in terms of the L_∞ gains of the individual mappings for each joint, namely $h_i: \tilde{r}_i^{-1} \to e_i$, where e_i is the position error for joint i. We rewrite the corresponding joint's error equation as,

$$k_v \dot{e}_i(t) + k_p e_i(t) = \tilde{r}_i^{-1}(t) \qquad (36)$$

where e_i, $\dot{e}_i$ and $\tilde{r}_i^{-1}$ are all time dependent signals. By taking the Laplace transform of (36), the transfer function of the error equation can be obtained as,

$$\frac{e_i(s)}{\tilde{r}_i^{-1}(s)} = h_i(s) = \frac{1}{k_v s + k_p} \tag{37}$$

The L_∞ gain[31] of the transfer function $h_i(s)$ can then be computed as $\|h_i\|_\infty = 1/k_p$. Now assuming that servo gains are chosen equal for all joints, L_∞ gain of the operator $\mathscr{H}$ can be directly written as,

$$\|\mathscr{H}\|_\infty = \frac{1}{k_p} \tag{38}$$

Next we look at the right hand side of (19). The nonlinear term $\tilde{R}^{-1}(\mathbf{z}(t))$ is in fact a time dependent vector transformation representing the error in the inverse dynamics modeling for each joint. That is,

$$\tilde{R}^{-1}(\mathbf{z}(t)) = \begin{bmatrix} \tilde{r}_1^{-1}(\mathbf{z}(t)) \\ \vdots \\ \tilde{r}_n^{-1}(\mathbf{z}(t)) \end{bmatrix}$$

The L_∞^n norm of $\tilde{R}^{-1}$ is then by,

$$\|\tilde{R}^{-1}(\mathbf{z}(t))\|_\infty = \max_i \sup_t |\tilde{r}_i^{-1}(t)| \tag{39}$$

If the bound on the approximation error of the ANN model for each joint is known, such that

$$\|r_i^{-1}(t) - \hat{r}_i^{-1}(t)\|_\infty = \|r_i^{-1}(t) - N_i\|_\infty = \|\tilde{r}_i^{-1}(t)\|_\infty \leq \varepsilon_i$$

holds, with ε_i being a non-negative real number representing the approximation error bound for joint i, then the bound on $\tilde{R}^{-1}$ can be written as,

$$|\tilde{R}^{-1}(\mathbf{z})\|_\infty = \max_i \varepsilon_i = \varepsilon \tag{41}$$

where ε denotes the maximum of individual joint approximation errors. This leads to the following Theorem.

Theorem 3.1 *If the error bounds on the approximation of the manipulator's inverse dynamic model by the ANN models satisfy,*

$$\|\tilde{R}^{-1}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\|_\infty = \|\tilde{R}^{-1}(\mathbf{z}(t))\|_\infty \leq \varepsilon$$

for some $\varepsilon \geq 0$, then the trajectory tracking error bounds on $\mathbf{e}$ and $\dot{\mathbf{e}}$ *are given by,*

$$\|\boldsymbol{e}\|_\infty \leq \frac{\varepsilon}{k_p} \tag{42}$$

$$\|\dot{\mathbf{e}}\|_\infty \leq \frac{2\varepsilon}{k_v} \tag{43}$$

where k_p and k_v denote position and velocity gains of each joint, respectively.

Proof:
The first inequality can be shown to hold directly, since it is based on the mapping $\mathscr{H}: \tilde{R}^{-1} \to \mathbf{e}$. Using the L_∞ gain of the operator $\mathscr{H}$ given by (38) and the bound on $\tilde{R}^{-1}(\mathbf{z}(t))$, we get,

$$\|\mathbf{e}\|_\infty \leq \|\mathscr{H}\|_\infty \|\tilde{R}^{-1}(\mathbf{z}(t))\|_\infty \tag{44}$$

$$\|\mathbf{e}\|_\infty \leq \frac{1}{k_p} \|\tilde{R}^{-1}(\mathbf{z}(t))\|_\infty \tag{45}$$

$$\|\mathbf{e}\|_\infty \leq \frac{\varepsilon}{k_p} \tag{46}$$

Using the error dynamics equations (19) and the bound on $\mathbf{e}$, the bound on $\dot{\mathbf{e}}$ can be obtained as follows,

$$\dot{\mathbf{e}} = -K_v^{-1} K_p \mathbf{e} + K_v^{-1} \tilde{R}^{-1}(\mathbf{z}) \tag{47}$$

taking the norm of both sides of (47) and using the triangle inequality of the norms, we get,

$$\|\dot{\mathbf{e}}\|_\infty \leq \|-K_v^{-1} K_p\|_i \|\mathbf{e}\|_\infty + \|K_v^{-1}\|_i \|\tilde{R}^{-1}(Z)\|_\infty \tag{48}$$

where $\|\cdot\|_i$ denotes the induced matrix norm in the L_∞ sense. Since K_v^{-1} and K_p are diagonal matrices with diagonal elements $1/k_v$ and k_p, respectively, induced matrix norms can simply be written as,

$$\|-K_v^{-1} K_p\|_i = \frac{k_p}{k_v} \quad \text{and} \quad \|K_v^{-1}\|_i = \frac{1}{k_v} \tag{49}$$

Using (49) and replacing $\|\mathbf{e}\|_\infty$ in (48) by (46) leads to,

$$\|\dot{\mathbf{e}}\|_\infty \leq \frac{\varepsilon}{k_v} + \frac{\varepsilon}{k_v} \tag{50}$$

$$\|\dot{\mathbf{e}}\|_\infty \leq \frac{2\varepsilon}{k_v} \tag{51}$$

This completes the proof. ■

This theorem basically defines the bounds on the tracking errors for a certain level of approximation of inverse dynamics by the ANN model structure. Next we present some simulation results for trajectory tracking and adaptation tests.

4 SIMULATION RESULTS

In order to evaluate various performance measures of the controller architecture, the proposed scheme is tested on a robotic manipulator model using simulation techniques. To verify the results given in Section 3, the overall scheme is simulated on a digital computer.

A two degrees of freedom (d.o.f.) robotic manipulator is simulated in order to demonstrate the various features of the controller algorithm. Gravity effects are *not* compensated throughout the simulation experiments except for the PD control experiments for comparison tests. Dynamic equations of a two link manipulator based on the simple mass distribution assumption are given as follows, and the derivation can be found in any robotics text.[1]

$$\tau_1 = (l_1^2(m_1+m_2) + m_2 l_2^2 + 2m_2 l_1 l_2 c_2)\ddot{q}_1 + (m_2 l_2^2 + m_2 l_1 l_2 c_2)\ddot{q}_2 - 2m_2 l_1 l_2 s_2 \dot{q}_1 \dot{q}_2 - m_2 l_1 l_2 s_2 \dot{q}_2^2 + (m_1+m_2) g l_1 c_1 + m_2 g l_2 c_{12} + v_1 \dot{q}_1 \quad (52)$$

$$\tau_2 = (m_2 l_2^2 + m_2 l_1 l_2 c_2)\ddot{q}_1 + l_2^2 m_2 \ddot{q}_2 + l_1 l_2 m_2 s_2 \dot{q}_1^2 + m_2 g l_2 c_{12} + v_2 \dot{q}_2 \quad (53)$$

τ_1 and τ_2 denote the torques at the first and the second joints, respectively and v_1 and v_2 are the viscous friction coefficients. The model parameters are chosen similar to the parameters of an experimental SCARA type manipulation.[32] Parameters are set to $m_1 = 10$ kg, $m_2 = 8$ kg, $l_1 = l_2 = 0.5$ meters, $v_1 = v_2 = 3.0$ N-M sec./rads, $g = 9.8$ kg·m/sec^2. A fourth order Runge-Kutta algorithm with a step size of $h = 0.005$ is used for the simulation.

Before any manipulator movement, the ANN model has no a priori information on manipulator dynamics, therefore the adjustable weights of each model are set to very small random numbers which are close to zero. This means, as the manipulator starts moving it has initially just the PD control driving its actuators. The feedforward terms start building up as more and more movements are made.

In all the simulation tests, a joint's desired position trajectory is chosen as,

$$\theta_d = a_0 + b_0 \sin(t) + b_0 \sin(2t) \quad (54)$$

with the desired trajectories for the velocity and acceleration being obtained by simply taking the first and the second derivatives of the position trajectory. The duration of the trajectory is chosen as 4 seconds, with $a_0 = 0.5$ and $b_0 = 0.2$.

4.1 Trajectory following and adaptation

Trajectory following tests: With the desired trajectory given by (54) for both joints, position (k_p) and velocity (k_v) feedback gains are set to $k_p = 625$ and $k_v = 125$ for both joints. These gains would result in overdamped error dynamics with a bandwidth equal to 25 rad/sec, when a close inverse dynamics approximation is achieved. Setting these gains are instrumental for determining the upper bounds for the tracking errors as it was shown in the previous section. However large feedback gains would cause instability in the adaptation phase of the controller.

As discussed in Section 2, there is **not** yet a **well-defined** procedure for the selection of an optimum ANN architecture for a given problem. The common approach is to try a few architectures that would give the required approximation while keeping the complexity of the network at a low level. A four layer network is chosen for our case. The learning rate α and the momentum term μ in the update equation are chosen as $\alpha = 0.0005$ and $\mu = 0.5$ for all the simulations. One ANN model is used at each joint as the adaptive feedforward unit as illustrated in Figure 2.

Figures 3 and 4 display the tracking position errors for the 1st, 14th and the 25th runs of the controller, respectively, along with the PD control outputs with no gravity compensation. Note the significant improvement in the tracking performance mainly due to the close approximation of the inverse dynamics by the ANN models. Figure 5 displays the position *RMS* errors for both joints plotted against the number of trial runs. As seen from the Figure 5, *RMS* position errors drop significantly after only a few trial runs, then reach an asymptotic value after around 20 runs.

For a comparison of performance, RMS, final position and peak position errors are tabulated for a PD controller (with gravity compensation) that used the same servo gains and the ANN based adaptive controller in Table I.

An important result which shows the effectiveness of the proposed scheme comes from the observation of the feedforward torque profiles for each joint. We monitored the torques generated at the outputs of the ANN models during the 25th run of the controller, and compared these torque profiles with the torques generated by the actual dynamic model of the manipulator (i.e. these represent the desired torque profiles). This experiment

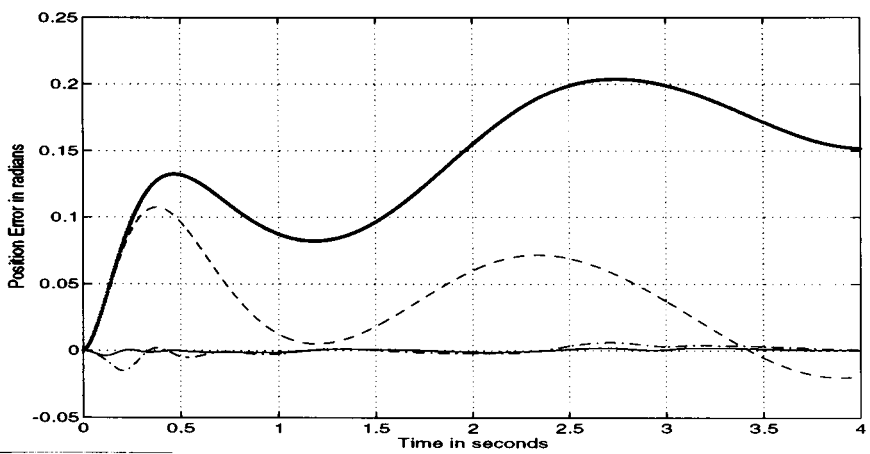

Fig. 3. Trajectory position errors of the PD Control (· · ·) and the 1st (– – –), 14th (– · – · –) and 25th (——) runs of the ANN controller for the 1st joint.

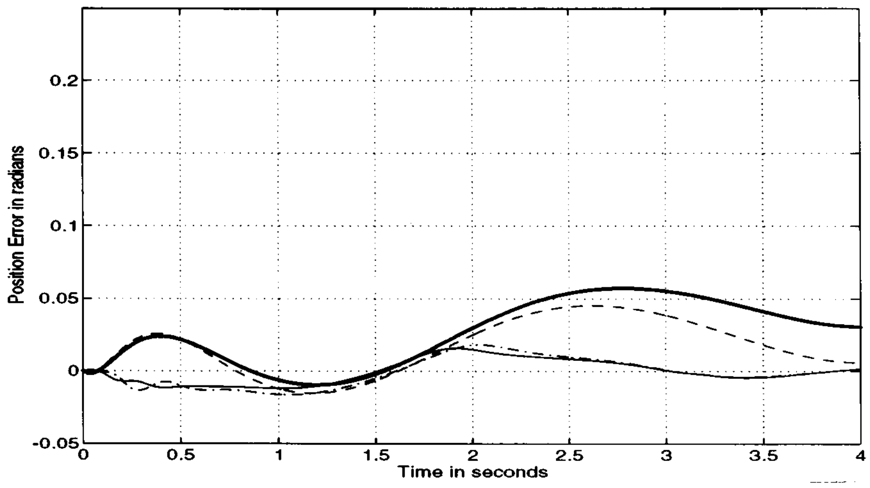

Fig. 4. Trajectory position errors of the PD Control (· · ·) and the 1st (– – –), 14th (– · – · –) and 25th (——) runs of the ANN controller for the 2nd joint.

demonstrates the level of approximation of inverse dynamics for the specific desired trajectory. Figure 6 shows the desired torque profile compared with the torque profiles generated by the ANN model during the 1st, 14th and 25th runs of the controller for the first joint. Similar results are obtained for the second joint. As seen from this figure ANN models do a very job in approximating the manipulator dynamics. This close approximation accounts for the fact that very small tracking errors are observed after only a few trials of the controller algorithm.

Adaptation Properties of the Controller: In this section we demonstrate the adaptation properties of the scheme to sudden changes in the manipulator dynamics. The experimental set-up is the same as in the trajectory following experiments. At the 25th run of the proposed controller, during which a close inverse dynamics approximation is achieved for the specific trajectory, the second link mass m_2 is changed from 8 kgs. to 16 kgs. at the 2 second mark (a 100% change). The position errors are plotted for both joints in Figure 7. This case can be considered as a situation where the manipulator suddenly picks up a heavy load. As shown in Figure 7, the controller effectively reduces the sudden jumps in the position errors and brings the errors down approximately to their previous levels in about $1-1.5$ seconds. In order to demonstrate the changes in the manipulator's inverse dynamics due to the end effector mass change and the

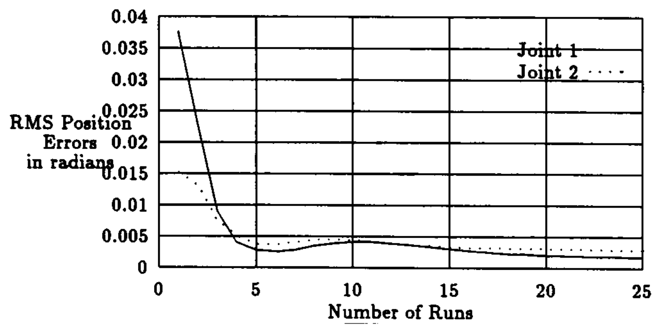

Fig. 5. RMS position errors for joint 1 (——) and joint 2 (· · ·) at each trial run of the ANN based controller.

ANN models' ability to track these changes, the torque profiles (i.e. the inverse dynamics) are monitored during the adaptation test. For the test when m_2 is changed from 8 kg to 16 kg, the desired torque profiles corresponding to this change and the torque profiles generated by the ANN models of each joint are plotted in Figure 8. The observed torque profiles converge to their desired values by the end of the trajectory. This convergence demonstrates the fast adaptation of the ANN weights to generate the required torque output. This effective adaptation eventually drives the position errors to their previous levels as illustrated in Figure 7. In order to check, if significant adaptation really occured in this short time interval, in the sense of significant changes in weight magnitudes, the Euclidean norm (L_2 norm) of the parameter vector **p** for the first joint's ANN model is computed and plotted in Figure 9 for that specific run. Note that the parameter vector norm shows a steep increase after the 2 second mark due to the sudden mass change and then converges in about $1-1.5$ seconds which correspond to $200-300$ adaptation steps for a step size of $h = 0.005$ sec. These simulation results are consistent with the analysis given in Section 3.3. When the system is operating at a nominal state, perturbations in system dynamics are compensated effectively by the adaptive structure of the controller.

5 DISCUSSION OF THE RESULTS AND CONCLUSION

The simulation experiments clearly demonstrated that the proposed architecture is an effective approach to the control of robotic manipulators with unknown dynamics. The proposed scheme is tested successfully for trajectory following and adaptation tasks.

Through simulation results, it's observed that better the approximation of the inverse dynamics for a given desired trajectory, the lower the tracking errors. Theorem 3.1 dictates that the bounds on the tracking errors are directly proportional to the bounds on the dynamics model approximation errors. This fact is actually demonstrated in the trajectory following test,

Table I. Performance comparison between a *PD* controller with *gravity compensation* and the proposed controller.

	PD Control		*ANN Based Control*	
Type of Error	*1st Link*	*2nd Link*	*1st Link*	*2nd Link*
Peak Position, pe_p	0.018 rad.	0.008 rad.	0.002 rad.	0.005 rad.
RMS Position, pe_{RMS}	0.011 rad.	0.004 rad.	0.0017 rad.	0.0028 rad.
Final Position, pe_f	0.013 rad.	0.006 rad.	0.020 rad.	0.0031 rad.

since very small position tracking errors are obtained when the generated torque profiles are closely matched with the system's inverse dynamics.

Another important result is the effective **adaptation capability** of the proposed scheme. It is shown that any change in the manipulator dynamics can be accounted for using the proposed scheme. This makes the use of the controller very attractive for real time applications where manipulator dynamics can experience sudden changes due to parameter variations, load changes and any possible external disturbances. When the end effector mass was doubled during the execution of a trajectory following task, the controller immediately acted to recover the sudden jumps in the tracking errors as shown in the Figure 7. The error recovery time which is about $1-1.5$ seconds is comparable with the recovery times of the more informed parameter based adaptive schemes.[4,5]

Simulations results are consistent with the local stability and convergence analysis given in Section 3.3.

Perhaps the most important advantage of the proposed controller is that it does **not** require a **parametric model** of the manipulator. This brings a high level of autonomy to the overall system. In its presented form, the controller utilizes one generic ANN model per joint. The proposed architecture requires no a priori knowledge of the system dynamics and approximates the inverse system dynamics for a specific trajectory following task. Data describing the mechanics of the manipulator become available only after movements have been processed. During this period of data acquisition and training period, the trajectory following performance will gradually improve. This is one of the major differences of the proposed approach compared to other ANN based techniques which utilize off-line training methods.[12,15,16]

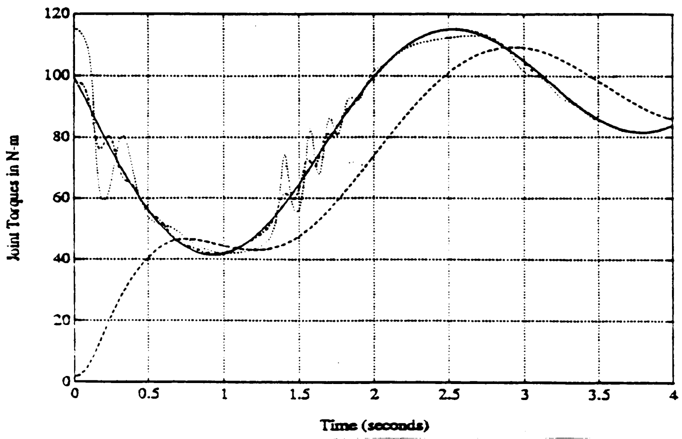

Fig. 6. The desired (——) and observed torque profiles during the 1st (– – –), 14th (· · ·) and 25th (– · –) runs of ANN controller for the 1st joint.

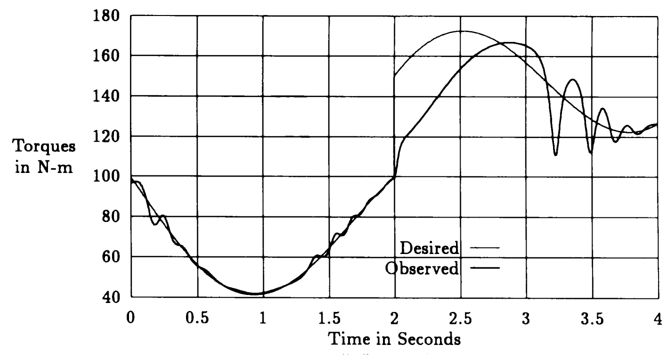

Fig. 8. The desired and observed torque profiles for the first link when m_2 is changed from 8 kg to 16 kg

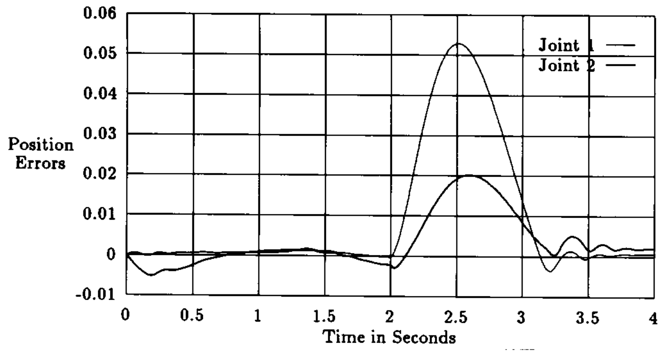

Fig. 7. Trajectory position errors for joint 1 and joint 2 (bold line), when the 2nd link mass m_2 is changed from 8 kgs to 16 kgs.

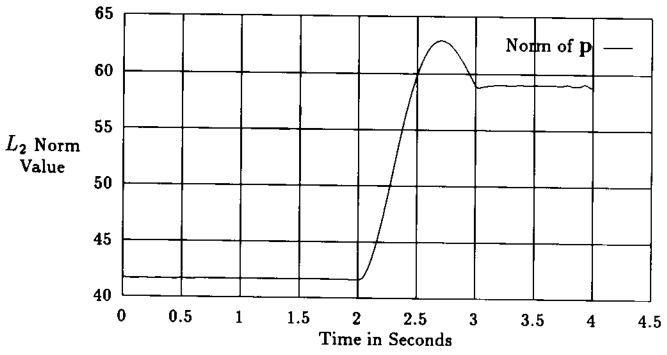

Fig. 9. Change of the Euclidian norm of all the weights of the ANN model during the adaptation experiment.

Table II. Performance comparison with other control schemes.

Controller Type	*1st Link*	*2nd Link*
PD Control (without gravity compensation)	0.1493 rad.	0.035 rad.
PD Control (gravity compensation)	0.011 rad.	0.004 rad.
Slotine & Li Adaptive Control	0.0011 rad.	0.0012 rad.
Off-line ANN training & control	0.027 rad.	0.026 rad.
Proposed ANN Controller	0.0017 rad.	0.0028 rad.

For comparison purposes, an **off-line ANN based traing** algorithm is tested using the prediction error learning method.[15] The ANN models for both joints have been trained for 25 runs to approximate the inverse dynamics of the manipulator. During this off-line training phase, only PD control has been used to drive the manipulator. Servo gains and all other ANN parameters were kept the same as in the previous tests. Nearly after 20 runs prediction error reached an asymptotic value and training was stopped. Using these ANN models as the feedforward elements along with the PD control, tracking errors were monitored. Additionally a more informed controller, a parametric adaptive controller based on Slotine and Li's method,[33] was also implemented. Again the same servo gains were used for a fair comparison and a large update rate was used for fast parameter convergence. The parameter adaptation was run for eight times over the same trajectory. Position RMS errors for these two alternative controllers are tabulated in Table II along with the RMS errors from our proposed controller and the PD control. Note that the off line training based ANN control didn ot perform as well as the proposed on line ANN controller, although it gave better results than the PD control without gravity compensation. An interesting result is that our proposed controller gave a similar tracking performance as the more informed parametric adaptive controller which assumed the a priori knowledge of the manipulator dynamics in parametric form.

As discussed in Section 3, a global stability analysis of the closed loop system is not trivial due to the coupled nonlinear dynamics of the update equation given in (31). However a local stability and convergence analysis of the closed loop system is possible[19] and it is summarized in Section 3.3. Finally, the use of the proposed controller is not restricted to manipulator control. It can be effectively utilized for any state feedback linearizable nonlinear system with unknown system dynamics.

Acknowledgements
The authors would like to thank anonymous *Robotica* reviewers for their constructive comments on the earlier version of the manuscript.

References
1. A.J. Koivo, *Fundamentals for Control of Robotic Manipulators* (New York John Wiley & Sons, 1989).
2. S. Dubowsky and D.T. DesForges, "The Application of Model-Referenced Adaptive Control to Robotic Manipulators" *J. Dynamic Systems, Measurement and Control* **101,** 193–200 (1979).
3. J. Craig, *Adaptive Control of Mechanical Manipulators* (Reading, Massachusetts, Addison-Wesley, 1989).
4. J.J. Craig, P. Hsu and S.S. Sastry, "Adaptive control of mechanical manipulators" *Int. J. Robotics Research* **6,** No. 2, 16–28 (1987).
5. J.E. Slotine and W. Li, "On the adaptive control of robot manipulators" *Int. J. Robotics Research* **6,** No. 3, 49–59 (1987).
6. R. Ortega and M.W. Spong, "Adaptive motion control of rigid robots" *Proc. IEEE 27th Conf. on Decision and Control,* Austin, Texas (1988) pp. 1575–1584.
7. S. Arimoto, S. Kawamura and F. Miyazaki, "Bettering operation of robots by learning" *J. Robotic Systems* **1,** No. 2, 123–140 (1984).
8. J.S. Albus, "A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC)" *J. Dynamic Systems, Measurement, and Control,* 220–227 (1975).
9. M.H. Raibert, "Motor Control and Learning by State Space Model" *AI-TR-439,* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977).
10. W.T. Miller, F.H. Glanz and L.G. Kraft, "Application of a learning algorithm to the control of robotic manipulators" *J. Robotics Research* **6,** No. 2, 84–98 (1987).
11. A.G. Barto, R.S. Sutton and C.W. Anderson, "Neuronlike adaptive elements that can solve difficult learning problems" *IEEE Trans. on Systems, Man and Cybernetics* **SMC-13,** 834–846 (1983).
12. D. Psaltis, A. Sideris and A.A. Yamamura, "A multilayer neural Network Controller" *IEEE Control Systems Magazine* 17–21 (April, 1988).
13. M. Kawato, "Computational schemes and neural network models for formation and control of multi-joint arm trajectory" *Neural Networks for Control* 197–228 (MIT Press, Cambridge, Mass., 1990).
14. A. Guez and J. Selinsky, "A trainable neuromorphic controller," *J. Robotic Systems* **5,** No. 4, 363–388 (1988).
15. K. Goldberg and B. Pearlmutter, "Using a Neural Network to learn the dynamics of the CMU Direct-Drive Arm II" *Technical Report No. CMU-CS-88-160* (Carnegie Mellon University, Pittsburgh, PA, 1988).
16. D.F. Bassi and G.A. Bekey, "High precision position control by cartesian trajectory feedback and connectionist inverse dynamics feedforward" *Proc. IEEE Int. Conf. on Neural Neural Networks,* Washington D.C. (1989) **Vol. 2,** pp. 325–331.
17. K. Ciliz and C. Işik, "Trajectory learning and control of robotic manipulators using Neural Networks" *IEEE International Symposium on Intelligent Control,* Fairfax, VA (1990) pp. 536–540.
18. K. Ciliz, "Artificial Neural Network Based Control of Nonlinear Systems With Application to Robotic Manipulators", *PhD thesis* (Syracuse University, Electrical Engineering Department, 1990).
19. K. Ciliz and C. Işik, "Stability and convergence of neurologic model based robotic controllers" *IEEE International Conference on Robotics and Automation,* Nice, France (May, 1992) pp. 2051–2056.

20. P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences" *PhD thesis* (Harvard University, 1974).
21. D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing*: *Explorations in the Microstructure of Cognition* (MIT Press, Cambridge, Mass., 1986).
22. A. Lapedes and R. Farber, "Nonlinear signal processing using Neural Networks: Prediction and system modeling" *Technical Report No. LA-UR-87-2662* (Los Alamos National Laboratory, 1987).
23. C. Işik and K. Ciliz, "A two level neural network system for learning control of robot motion" *Proc. IEEE Int. Symp. Intelligent Control,* Arlington, VA (August, 1988) pp. 519–522.
24. K. Ciliz and C. Işik, "Time optimal control of mobile robot motion using neural nets" *IEEE International Symposium on Intelligent Control,* Albany, N.Y. (1989) pp. 368–373.
25. R.P. Gorman and T.J. Sejnowski, "Learned classification of sonar targets using a massively parallel network" *IEEE Trans. on Acoustics, Speech and Signal Processing* **ASSP-36,** No. 7, 1135–1140 (1989).
26. K.C. Funahashi, "On the approximate realization of continuous mappings by Neural Networks" *Neural Networks* **2,** 183–192 (1989).
27. G. Cybenko, "Approximations by superpositions of a sigmoidal function" *Tech. Prep.,* (University of Illinois, Center for Supercomputer Research and Development, Urbana, IL, 1989).
28. K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators" *Neural Networks* **2,** 359–366 (1989).
29. K. Ciliz and K. Narendra, "Multiple model based adaptive control of robotic manipulators" *IEEE Conference on Decision and Control,* Lake Buena Vista, Florida (1994) pp. 1305–1310.
30. K. Ciliz and M. Tomizuka, "Modeling and compensation of frictional uncertainties in motion control: A neural network based approach" *American Control Conference* (June 1995).
31. C.A. Desoer and M. Vidyasagar, *Feedback Systems*: *Input–Output Properties* (New York, NY, Academic Press, 1975).
32. *Direct drive manipulator research and development package* (California, USA, 1992. Integrated Motions Incorporated).
33. J.E. Slotine and W. Li, "Adaptive manipulator control: A case study" *IEEE Trans. on Automatic Control* **33,** No. 11, 995–1003 (1988).